International Conference on Rewriting Techniques and Applications 2010 (Edinburgh), pp. 135-150
http://rewriting.loria.fr/rta/

A PROOF CALCULUS WHICH REDUCES SYNTACTIC BUREAUCRACY

ALESSIO GUGLIELMI [1] AND TOM GUNDERSEN [2] AND MICHEL PARIGOT [3]

[1] University of Bath and LORIA & INRIA Nancy–Grand Est

[2] LIX & INRIA Saclay–Île-de-France

[3] Laboratoire PPS, UMR 7126, CNRS & Université Paris 7
E-mail address: parigot@pps.jussieu.fr

Abstract. In usual proof systems, like the sequent calculus, only a very limited way of combining proofs is available through the tree structure. We present in this paper a logic-independent proof calculus, where proofs can be freely composed by connectives, and prove its basic properties. The main advantage of this proof calculus is that it allows to avoid certain types of syntactic bureaucracy inherent to all usual proof systems, in particular the sequent calculus. Proofs in this system closely reflect their atomic flow, which traces the behaviour of atoms through structural rules. The general definition is illustrated by the standard deep-inference system for propositional logic, for which there are known rewriting techniques that achieve cut elimination based only on the information in atomic flows.

1. Introduction

One of the biggest challenges we are facing in structural proof theory, especially when looking at computational interpretations of proof systems, is syntactic dependency: formalisms impose irrelevant constraints, typically an arbitrary order between operations that are in principle independent from each other. The first explicit attempts to lower this syntactic dependency, called 'bureaucracy', date back to the eighties, with the concept of proof net for linear logic: proof nets are geometric traces of sequent-calculus proofs, which eliminate some syntactic constraints. Proof nets have been widely studied in the past two decades. Despite being powerful tools, they have two obvious limitations: 1) they apply directly only to specific logics, 2) they are not 'deductive' and rely on the sequent calculus for the deducing task.

1998 ACM Subject Classification: F.4.2.

Key words and phrases: Logic, Proof theory, Deep Inference, Flow graphs, Proof Systems, Open Deduction, Rewriting, Confluence, Termination.

Proceedings of the 21st International Conference on Rewriting Techniques and Applications, Edinburgh, July, 2010
Editors: Christopher Lynch
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.RTA.2010.135

A typical example of two proofs in the sequent calculus that are 'morally' the same but syntactically different, only because rules are applied in a different order, is the following:

$$\dfrac{\dfrac{\dfrac{\vdash a, \bar{a} \quad \vdash b, \bar{b}}{\vdash a \wedge b, \bar{a}, \bar{b}}\wedge}{\vdash a \wedge b, \bar{a} \vee \bar{b}}\vee \qquad \vdash c, \bar{c}}{\vdash (a \wedge b) \wedge c, \bar{a} \vee \bar{b}, \bar{c}}\wedge \qquad\qquad \dfrac{\dfrac{\dfrac{\vdash a, \bar{a} \quad \vdash b, \bar{b}}{\vdash a \wedge b, \bar{a}, \bar{b}}\wedge \qquad \vdash c, \bar{c}}{\vdash (a \wedge b) \wedge c, \bar{a}, \bar{b}, \bar{c}}\wedge}{\vdash (a \wedge b) \wedge c, \bar{a} \vee \bar{b}, \bar{c}}\vee$$

This kind of bureaucracy is present in all the usual deduction formalisms. One could imagine that there are simple ways to remove it: for instance quotienting proofs by some equivalence relation. However, this would not work, in particular, because logical rules are, in general, not linear.

The approach we develop in this paper makes use of the deep-inference methodology. Deep inference is a deduction framework (see [Gug07, BT01, Brü04]), where deduction rules apply arbitrarily deep inside formulae, contrary to traditional proof systems like natural deduction and sequent calculus, where deduction rules only deal with their outermost structure. The main reason to use deep inference is that it provides more freedom in designing proof systems, while maintaining the proof theoretic properties of interest, first and foremost cut elimination.

The simple principle of allowing inference to happen inside formulae leads to a natural change in the underlying structure of proofs, where rules are *unary*: one premiss and one conclusion. While proofs in usual deduction systems take the asymmetric form of a tree, deep inference allows to have a symmetric closure operation along the top-down axis. While sequent-calculus proofs cannot be dualised by flipping, this is always possible in deep inference, and it logically corresponds to dualities like the De Morgan one in classical logic.

A general methodology allows to design deep-inference deduction systems having more symmetries and finer structural properties than the sequent-calculus ones. For instance, cut and identity become really dual of each other, whereas they are only morally so in the sequent calculus, and all structural rules can be reduced to their atomic form, whereas contraction can not in the sequent calculus [Brü03].

All usual logics have deep-inference deduction systems enjoying cut elimination (see [Gug] for a complete overview). The standard proof system for propositional classical logic in deep inference is system SKS [BT01, Brü04]. The traditional methods of cut elimination of the sequent calculus can be adapted to a large extent to deep inference, despite having to cope with a higher generality [BT01, Brü04]. New methods are also achievable, based on weak computational traces of proofs called *atomic flows*. Atomic flows are directed acyclic graphs extracted from proofs that can be equipped with rewrite rules representing cut-elimination. Though being very simple (they trace only structural rules and forget logical rules), they are strong enough to faithfully represent and control cut-elimination procedures [GG08, Gun09], even a surprising quasipolynomial one [BGGP09].

So far, these developments have taken place inside a specific deep-inference formalism, dubbed the *calculus of structures*, where proofs are *sequential*, i.e., chains of formulae that are nothing else than terms in a term-rewriting chain generated by applications of unary rules of a given proof system. This very simple setting is not particularly intuitive and suffers from some forms of 'syntactic bureaucracy', like the one described before, where an irrelevant order of the rules is imposed by the formalism: this can be immediately appreciated by looking at the following two different proofs (that we take as logically equivalent, in some

unspecified logic):

$$\dfrac{\dfrac{A \wedge B}{C \wedge B}}{C \wedge D} \quad \text{and} \quad \dfrac{\dfrac{A \wedge B}{A \wedge D}}{C \wedge D} \quad ;$$

this is an obvious case of independent rewriting on two terms, but the sequential notion of proof does not allow for a canonical proof.

This paper shows that we can do better. The situation where the formalism imposes an irrelevant order of application of two rules to two independent subformulae is called bureaucracy of type A [Gug04a, Str09]. We define here a new formalism, called *open deduction*, that contains the calculus of structures as a special case and that provides a wider universe of proofs, where it is possible to normalise proofs into proofs where bureaucracy of type A is absent, using a simple procedure which is confluent and terminating. We call the proofs in this normal form *synchronal*. Referring to the example provided before, we have that open deduction allows the proof

$$\dfrac{A}{C} \wedge \dfrac{B}{D} \quad .$$

In fact, the definition of open deduction is based on a very simple, alternative but equivalent deep-inference notion to the term-rewriting one of the calculus of structures: proofs can be composed by connectives.

It should be emphasised that open deduction is a *logic-independent formalism*, which applies to all usual logics, thanks to its deep-inference foundation. Even if we are working at a high level of abstraction, we can still prove meaningful properties. In section 2, we exhibit a simple rewrite procedure that is confluent and terminating and that allows to transform any derivation into one in synchronal form, which is, moreover, of smaller size.

Of course, a natural question that pops up is: what happens to cut elimination? In particular, can we generalise the technique of atomic flows, that has been used in the particular case of the SKS system for classical propositional logic, to open deduction? We provide the basis of a positive answer in section 3. Thanks to deep inference, which allows us to represent logics with rules that are either *atomic* or *linear*, we define a general notion of atomic flows for open deduction. We show that the bureaucracy-elimination procedure of section 2, which transforms any open-deduction derivation into a synchronal one, has an important property: atomic flows are invariant under it.

In section 4, we restrict to system SKS for classical propositional logic and show that the rewrite rules of atomic flows, which are known to be sound with respect to sequential and synchronal derivations, are also sound with respect to open deductions in general. This means, in particular, that open-deduction cut elimination can be controlled by atomic flows the same way calculus-of-structures cut elimination is.

2. The Open Deduction Formalism

In this section, we present the open deduction formalism in a logic-independent way and illustrate it with the standard formalisation of propositional classical logic in deep inference. We define a canonical form of open deductions, called *synchronal*, which is free of the bureaucracy of type A described in the introduction. We prove that there is a

simple confluent and terminating rewriting procedure which transforms an arbitrary open deduction into a synchronal one.

Definition 2.1. We have the following mutually disjoint, countable sets:

- the set of *atoms* $\mathcal{A}$, whose elements are denoted by a, b, c and d (possibly with subscripts);
- for each $m < \omega$ and $n < \omega$, the set $\mathcal{R}_{m,n}$ of *logical relations* of *positive arity* m and *negative arity* n; we denote by $\mathcal{R}$, the set $\bigcup_{m,n\geq 0} \mathcal{R}_{m,n}$ of all logical relations; the elements of $\mathcal{R}$ are denoted by r (possibly with subscripts) and dedicated symbols for usual logical relations; the logical relations of positive and negative arity 0 are called *logical constants*.

Comment 1. The intended meaning of the positive and negative arities is that the following holds in the deduction system under consideration: if r is a logical relation of $\mathcal{R}_{m,n}$ and for each $i \leq n + m$, the formula B_i is deducible from the formula A_i then $r(B_1, \ldots, B_n, A_{n+1}, \ldots, A_{n+m})$ is deducible from $r(A_1, \ldots, A_n, B_{n+1}, \ldots, B_{n+m})$. The definition of derivation we will take in this paper ensures this property.
The connectives $\wedge$ and $\vee$ of classical logic are in $\mathcal{R}_{2,0}$, $\rightarrow$ is in $\mathcal{R}_{1,1}$ and $\neg$ is in $\mathcal{R}_{0,1}$.
It should be noted that there are also connectives of classical logic that do not satisfy the required property, for instance $\leftrightarrow$, but they can always be defined from connectives that do.

Definition 2.2. Let a set of logical relations $R = \bigcup_{m,n\geq 0} R_{m,n}$, where $R_{m,n} \subseteq \mathcal{R}_{m,n}$, be given.

(1) The set $\mathcal{F}_{\mathcal{R}}$ of *formulae*, denoted by A, B, C and D (possibly with subscripts), is defined inductively by:
   (a) $\mathcal{A} \subseteq \mathcal{F}_R$;
   (b) $\mathcal{F}_R$ is closed by *logical relation composition*: if $r \in R_{m,n}$ and $A_1, \ldots, A_{m+n} \in \mathcal{F}_R$, then $r(A_1, \ldots, A_{m+n}) \in \mathcal{F}_R$.

(2) The set $\mathcal{D}_R$ of *prederivations*, denoted by Φ and Ψ (possibly with subscripts), is defined inductively by :
   (a) $\mathcal{A} \subseteq \mathcal{D}_R$;
   (b) $\mathcal{D}_R$ is closed by *logical relation composition*: if $r \in R_{m,n}$ and $\Phi_1, \ldots, \Phi_{m+n} \in \mathcal{D}_R$, then $r(\Phi_1, \ldots, \Phi_{m+n}) \in \mathcal{D}_R$; and
   (c) $\mathcal{D}_R$ is closed by *inference composition*: if $\Phi_1, \Phi_2 \in \mathcal{D}_R$ then $\dfrac{\Phi_1}{\Phi_2} \in \mathcal{D}_R$.

   Inference composition is supposed to be associative.

(3) The *premiss* and *conclusion* functions $\mathsf{pr}, \mathsf{cn} : \mathcal{D}_R \rightarrow \mathcal{F}_R$ are defined inductively as follows:
   (a) if $\Psi \in \mathcal{A}$, then $\mathsf{pr}\,\Psi = \mathsf{cn}\,\Psi = \Psi$;
   (b) if $r \in R$ and $\Psi = r(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n)$, then
   $$\mathsf{pr}\,\Psi = r(\mathsf{pr}\,\Phi_1, \ldots, \mathsf{pr}\,\Phi_m, \mathsf{cn}\,\Phi'_1, \ldots, \mathsf{cn}\,\Phi'_n) \quad \text{and}$$
   $$\mathsf{cn}\,\Psi = r(\mathsf{cn}\,\Phi_1, \ldots, \mathsf{cn}\,\Phi_m, \mathsf{pr}\,\Phi'_1, \ldots, \mathsf{pr}\,\Phi'_n) \quad ; \text{and}$$
   (c) if $\Psi = \dfrac{\Phi_1}{\Phi_2}$, then $\mathsf{pr}\,\Psi = \mathsf{pr}\,\Phi_1$ and $\mathsf{cn}\,\Psi = \mathsf{cn}\,\Phi_2$.

(4) The sets of *positive contexts* and *negative contexts* are defined inductively as follows:
   (a) $\{\ \}$ is a positive context;

(b) if $r \in R_{m,n}$, $k \leq m$, A_k is a positive (resp. negative) context and for each $i \neq k$, A_i is a formula, then $r(A_1, \ldots, A_{m+n})$ is a positive (resp. negative) context;
(c) if $r \in R_{m,n}$, $m < k \leq m+n$, A_k is a positive (resp. negative) context and for each $i \neq k$, A_i is a formula, then $r(A_1, \ldots, A_{m+n})$ is a negative (resp. positive) context.

Contexts are denoted $K\{\ \}$. We use $K^+\{\ \}$ and $K^-\{\ \}$ when we need to specify the polarity of the context.

The *size* $|\Psi|$ of a prederivation (or formula or context) Ψ is the number of occurrences of atoms and logical relations in it.

Comment 2. Prederivations in open deduction have a natural planar representation where the inference composition is represented vertically and the logical relation composition is represented horizontally (see example 2.4)

Notation 1. For typographic convenience, inference composition of two prederivations Φ_1 and Φ_2 is also denoted by $\Phi_1|\Phi_2$. A prederivation Φ with $\mathsf{pr}\,\Phi = A$ and $\mathsf{cn}\,\Phi = B$ is denoted $\Phi : A \to B$ and represented in figures by

$$\begin{array}{c} A \\ \Phi\| \\ B \end{array} .$$

Example 2.3. Classical propositional logic.

- The *logical relations* are:
  - *disjunction* $\vee$ and *conjunction* $\wedge$ which are in $\mathcal{R}_{2,0}$;
  - *negation* $\neg$ is which is in $\mathcal{R}_{0,1}$;
  - *logical constants*, f (false) and t (true), which are in $\mathcal{R}_{0,0}$.
- In the usual presentation of classical propositional logic, the SKS system, negation is not taken as a primitive connective, but defined by duality from its atomic case. The negation of an atom a is denoted $\bar{a}$. The disjunction and conjunction of two formulae A and B are denoted respectively $[A \vee B]$ and $(A \wedge B)$: the different brackets have the only purpose of improving legibility. We usually omit external brackets of formulae and sometimes we omit superfluous brackets under associativity. Example of formulae are $b \wedge [a \vee c]$ and $\neg [a \vee b] \wedge [a \vee c]$. An example of context $K\{\ \}$ is $b \wedge [\{\ \} \vee c]$; in this case $K\{a\}$ is $b \wedge [a \vee c]$, $K\{b\}$ is $b \wedge [b \vee c]$ and $K\{a \wedge d\}$ is $b \wedge [(a \wedge d) \vee c]$.

Example 2.4. The prederivation

$$\left(\dfrac{\dfrac{a_1}{a_2}}{a_3} \wedge a_4 \right) \vee \neg \left[\dfrac{\left(\dfrac{a_6}{a_5} \wedge a_7 \right)}{a_8} \vee a_9 \right] .$$

has $(a_1 \wedge a_4) \vee \neg [a_8 \vee a_9]$ as premiss and $(a_3 \wedge a_4) \vee \neg [(a_6 \wedge a_7) \vee a_9]$ as conclusion.

Notation 2. If $K\{\ \}$ is a context and Φ a prederivation, we denote by $K\{\Phi\}$ the prederivation obtained by putting Φ in place of the hole in $K\{\ \}$. For example,

if $K\{\ \}$ is $b \wedge [\{\ \} \vee c]$ and Φ is $\left(\dfrac{a_6}{a_5} \wedge a_7 \right)$, then $K\{\Phi\}$ is $b \wedge \left[\left(\dfrac{a_6}{a_5} \wedge a_7 \right) \vee c \right]$.

Definition 2.5. Given two prederivations $\Phi_1 : A \to B$ and $\Phi_2 : B \to C$, the *composition* of Φ_1 and Φ_2 , denoted $\Phi_1; \Phi_2 : A \to C$, is a prederivation defined inductively as follows:

- if $\Phi_1 \in \mathcal{A}$ then $\Phi_1; \Phi_2 \;=\; \Phi_2$,
- if $\Phi_1 \;=\; \dfrac{\Phi_1'}{\Phi_1''}$ then $\Phi_1; \Phi_2 \;=\; \dfrac{\Phi_1'}{\Phi_1''; \Phi_2}$
- if $\Phi_1 \;=\; r(\Phi_1^1, \ldots, \Phi_n^1, \Phi_{m+1}^1, \ldots \Phi_{m+n}^1)$ with $r \in R_{n,m}$, then
  - if $\Phi_2 \;=\; \dfrac{\Phi_2'}{\Phi_2''}$ then $\Phi_1; \Phi_2 \;=\; \dfrac{\Phi_1; \Phi_2'}{\Phi_2''}$
  - if $\Phi_2 \;=\; r(\Phi_1^2, \ldots, \Phi_n^2, \Phi_{n+1}^2, \ldots \Phi_{n+m}^2)$ then
    $$\Phi_1; \Phi_2 \;=\; r(\Phi_1^1; \Phi_1^2, \ldots, \Phi_n^1; \Phi_n^2, \Phi_{n+1}^2; \Phi_{n+1}^1, \ldots, \Phi_{n+m}^2; \Phi_{n+m}^1)$$

Lemma 2.6. *The composition of two prederivations is well defined: the definition is compatible with associativity of inference composition. Moreover, though being given by an asymmetric double induction, the composition of two prederivations Φ_1 and Φ_2 is symmetric in the sense that:*

- $\Phi; a = a; \Phi = \Phi$ *(and more generally $\Phi; A = A; \Phi = \Phi$); and*
- $(\Phi_1|\Phi_2); \Psi = \Phi_1|(\Phi_2; \Psi)$ *and* $\Phi; (\Psi_1|\Psi_2) = (\Phi; \Psi_1)|\Psi_2$.

Notation 3. For typographic convenience, composition of two prederivations $\Phi_1 \colon A \to B$ and $\Phi_2 \colon B \to C$ is represented in figures by

$$\begin{array}{c} \Phi_1 \\ \vdots \\ \Phi_2 \end{array} \quad .$$

Lemma 2.7. $|\Phi_1; \Phi_2| = |\Phi_1| + |\Phi_2| - |\,\mathsf{cn}\,\Phi_1|$.

Lemma 2.8. *Composition of prederivations is associative.*

Proposition 2.9. *Given any two prederivations Φ and Ψ, we have:* $\Phi|\Psi = \Phi; (\mathsf{cn}\,\Phi|\,\mathsf{pr}\,\Psi); \Psi$.

Proof. By the previous lemmas, we have:
$\Phi|\Psi = (\Phi; \mathsf{cn}\,\Phi)|(\mathsf{pr}\,\Psi; \Psi) = \Phi; (\mathsf{cn}\,\Phi|(\mathsf{pr}\,\Psi; \Psi)) = \Phi; ((\mathsf{cn}\,\Phi|\,\mathsf{pr}\,\Psi); \Psi) = \Phi; (\mathsf{cn}\,\Phi|\,\mathsf{pr}\,\Psi); \Psi$. ∎

Comment 3. The previous proposition states an important property: any prederivation can be 'decomposed' in such a way that inference composition only applies to formulae.

Definition 2.10. An *basic inference step* ρ is an inference composition $\dfrac{A}{B}$, where A and B are formulae called *premiss* and *conclusion*, respectively: it is denoted $\rho\,\dfrac{A}{B}$ or $A|_\rho B$. In concrete deduction systems, the set of basic inference steps is generated by a (finite) set of *inference rules* (with names for arbitrary atoms and formulae) from which the inference steps are instances.

If ρ is a basic inference step $\dfrac{A}{B}$, then the *flipped basic inference step* $\dfrac{B}{A}$ is denoted $\rho^\perp$, where $\cdot^\perp$ is an involution on the set of inference steps, i.e. $(\rho^\perp)^\perp = \rho$. If S is a set of basic inference steps, then $S^\perp$ is the set of basic inference steps $(\rho^\perp)$ for $\rho \in S$.

If $K\{\ \}$ is a positive (resp. negative) context and ρ is the basic inference step $\dfrac{A}{B}$, then we denote by $K\{\rho\}$ the *inference step* $\dfrac{K\{A\}}{K\{B\}}$ (resp. $\dfrac{K\{B\}}{K\{A\}}$).

The concept of derivation is obtained from the one of prederivation by restricting the inference composition to given inference steps.

Definition 2.11. Let a set of relations R and a set of basic inference steps S be given. The set $\mathcal{D}_{R,S}$ of *S-derivations* is defined inductively by

(1) $\mathcal{A} \subseteq \mathcal{D}_R$;
(2) if $r \in R_{m,n}$ and $\Phi_1, \ldots, \Phi_{m+n} \in \mathcal{D}_R$, then $\Psi = r(\Phi_1, \ldots, \Phi_{m+n}) \in \mathcal{D}_R$;
(3) if $\Phi_1, \Phi_2 \in \mathcal{D}_{R,S}$ then $\dfrac{\Phi_1}{\Phi_2} \in \mathcal{D}_{R,S}$ if there exists a context $K\{\ \}$ and a basic inference step ρ from S such that $\dfrac{\mathsf{cn}\,\Phi_1}{\mathsf{pr}\,\Phi_2}$ is the inference step $K\{\rho\}$.

Notation 4. An inference composition of Φ_1 and Φ_2 is denoted ${}_{K\{\rho\}}\dfrac{\Phi_1}{\Phi_2}$ or $\Phi_1|_{K\{\rho\}}\Phi_2$, where $K\{\rho\}$ is the inference step $\dfrac{\mathsf{cn}\,\Phi_1}{\mathsf{pr}\,\Phi_2}$. A S-derivation $\Phi : A \to B$ is denoted $\begin{matrix} A \\ \Phi\|S \\ B \end{matrix}$.
When it is clear from the context, we use the term *derivations* instead of *S-derivations* and omit S from the notation.

Definition 2.12. We define two canonical forms of *S-derivations*:

(1) the set of *sequential S-derivations* is the set of S-derivations where, in case (2) of Definition 2.11, $\Phi_1, \ldots, \Phi_{n+m}$ are formulae.
(2) the set of *synchronal S-derivations* is the set of S-derivations where, in case (3) of Definition 2.11, $K\{\ \} = \{\ \}$, i.e. inference composition is restricted to basic inference steps.

Comment 4. Sequential derivations are the usual derivation of the calculus of structures. Synchronal derivations are derivations which are free of the type A bureaucracy described in the introduction (see example 2.15).

Lemma 2.13. *If Φ_1 and Φ_2 are S-derivations (resp. sequential S-derivations, synchronal S-derivations), then $\Phi_1; \Phi_2$ is a S-derivation (resp. sequential S-derivation, synchronal S-derivation).*

Example 2.14. The system SKS for classical propositional logic.
The set S of basic inference steps is the set of instances of the inference rules given below. The (usual) deep inference derivations of SKS are the sequential S-derivations.

Structural inference rules:

$$\mathsf{ai}{\downarrow}\,\frac{\mathsf{t}}{a \vee \bar{a}} \qquad \mathsf{aw}{\downarrow}\,\frac{\mathsf{f}}{a} \qquad \mathsf{ac}{\downarrow}\,\frac{a \vee a}{a}$$

identity (interaction) *weakening* *contraction*

$$\mathsf{ai}{\uparrow}\,\frac{a \wedge \bar{a}}{\mathsf{f}} \qquad \mathsf{aw}{\uparrow}\,\frac{a}{\mathsf{t}} \qquad \mathsf{ac}{\uparrow}\,\frac{a}{a \wedge a}$$

cut (cointeraction) *coweakening* *cocontraction*

,

Logical inference rules:

$$\mathsf{s}\,\frac{A \wedge [B \vee C]}{(A \wedge B) \vee C} \qquad \mathsf{m}\,\frac{(A \wedge B) \vee (C \wedge D)}{[A \vee C] \wedge [B \vee D]}$$

switch *medial*

.

In addition to these two rules, there are *equality* rules $=\dfrac{C}{D}$, for C and D in opposite sides in one of the following equations:

$$\begin{array}{ll} A \vee B = B \vee A & A \vee \mathsf{f} = A \\ A \wedge B = B \wedge A & A \wedge \mathsf{t} = A \\ [A \vee B] \vee C = A \vee [B \vee C] & \mathsf{t} \vee \mathsf{t} = \mathsf{t} \\ (A \wedge B) \wedge C = A \wedge (B \wedge C) & \mathsf{f} \wedge \mathsf{f} = \mathsf{f} \end{array} \quad . \tag{2.1}$$

Comment 5. The SKS system shows an important property of deep inference formalism that we will use in the next section. The rules are of one of the two following kinds:

- *atomic rules*: only one atom name appears and no formula name appears.
- *linear rules*: each atom or formula name which appears in the premiss (resp. conclusion) appears exactly once in the conclusion (resp. premiss)

Example 2.15. We give here an example of a sequential derivation in SKS and its corresponding synchronal form.

$$\mathsf{ac}{\downarrow}\,\frac{\mathsf{m}\,\dfrac{\mathsf{m}\,\dfrac{\mathsf{ac}{\downarrow}\,\dfrac{\mathsf{m}\,\dfrac{(a \wedge b \wedge c) \vee (a \wedge b \wedge c)}{[(a \wedge b) \vee (a \wedge b)] \wedge [c \vee c]}}{[(a \wedge b) \vee (a \wedge b)] \wedge c}}{[a \vee a] \wedge [b \vee b] \wedge c}}{a \wedge [b \vee b] \wedge c}}{a \wedge b \wedge c} \qquad \mathsf{m}\,\frac{(a \wedge b \wedge c) \vee (a \wedge b \wedge c)}{\mathsf{m}\,\dfrac{(a \wedge b) \vee (a \wedge b)}{\dfrac{a \vee a}{a} \wedge \dfrac{b \vee b}{b}} \wedge \dfrac{c \vee c}{c}}$$

One can see on this example that the size of the synchronal derivation is smaller than the size of the sequential one.

Definition 2.16. (1) A *synchronisation redex* is an inference composition $K\{\rho\}\,\dfrac{\Phi_1}{\Phi_2}$ where $K\{\ \} \neq \{\ \}$. Note that thanks to proposition 2.9, a synchronisation redex can always be written $\Phi_1; (\mathsf{cn}\,\Phi_1|_{K\{\rho\}}\,\mathsf{pr}\,\Phi_2); \Phi_2$.
(2) The *contractum* of a synchronisation redex $\Phi_1; (\mathsf{cn}\,\Phi_1|_{K\{\rho\}}\,\mathsf{pr}\,\Phi_2); \Phi_2$ is $\Phi_1; K\{C|_\rho D\}; \Phi_2$, where C and D are the premiss and conclusion of the inference step ρ.

(3) The *synchronisation* reduction $\overset{\mathsf{sync}}{\to}$ on $\mathcal{D}_{\mathcal{R},S}$ is defined by: $\Phi \overset{\mathsf{sync}}{\to} \Psi$ iff Ψ is obtained from Φ by replacing a synchronisation redex by its contractum.

Lemma 2.17. *If Φ is a redex and Ψ its contractum, then $|\Psi| < |\Phi|$.*

Proof. Let $\Phi = \Phi_1; (\mathsf{cn}\, \Phi_1|_{K\{\rho\}}\, \mathsf{pr}\, \Phi_2); \Phi_2$ be a redex and $\Psi = \Phi_1; K\{C|_\rho D\}; \Phi_2$ its contractum. We have $|K\{C|_\rho D\}| = |K\{\ \}| + |C| + |D|$. By Lemma 2.7 we then have

$$|\Psi| = |\Phi_1| + |K\{\ \}| + |C| + |D| + |\Phi_2| - |K\{C\}| - |K\{D\}| = |\Phi_1| + |\Phi_2| - |K\{\ \}| < |\Phi|$$

■

Theorem 2.18. *The synchronisation reduction is confluent and terminating. Moreover, each derivation reduces in a number of steps less than its size to its normal form which is a synchronal derivation.*

Proof. $\overset{\mathsf{sync}}{\to}$ is terminating because of Lemma 2.17. We show that $\overset{\mathsf{sync}}{\to}$ is locally confluent. Consider a derivation Ψ with two synchronisation redexes r_1 and r_2. Let Φ the smallest subderivation of Ψ which contains r_1 and r_2. There are two cases:
(1) $\Phi = r(\Phi_1, \ldots, \Phi_i, \ldots, \Phi_j, \ldots, \Phi_1)$ with r_1 in Φ_i, r_2 in Φ_j and $i \neq j$. Then the order of reduction of the two redexes obviously doesn't matter.
(2) $\Phi = \Phi_1|_\rho \Phi_2$. Then one of the following holds:

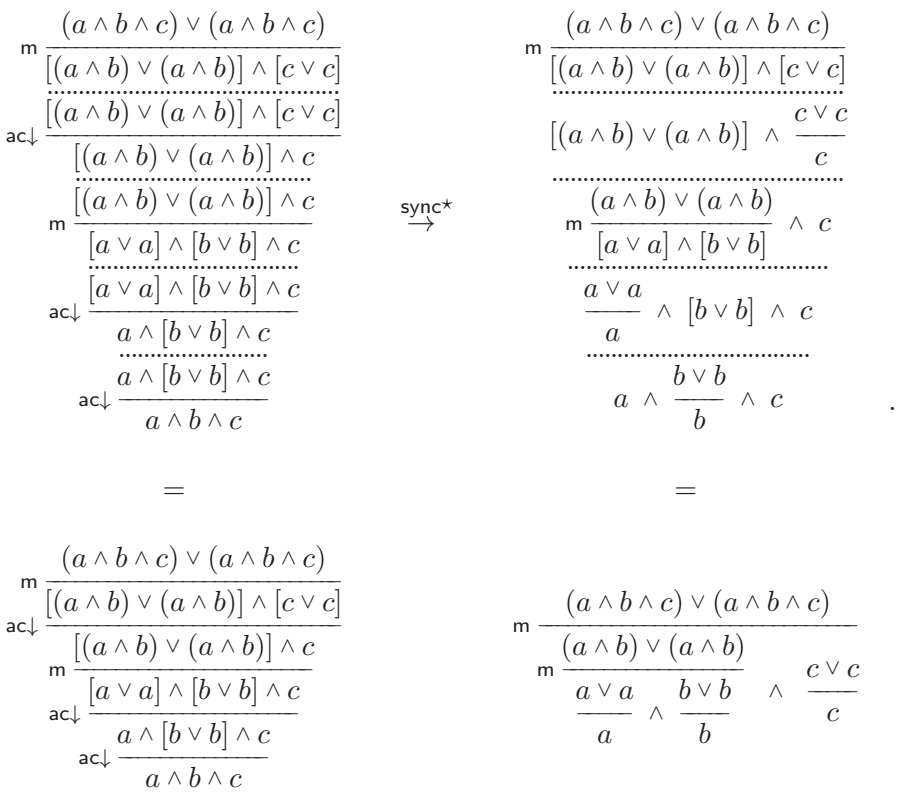
- One redex is in Φ_1 and the other in Φ_2. Then the order of reduction of the two redexes obviously doesn't matter.
- Φ is one of the redexes and the other one is in Φ_1 or Φ_2. Suppose for example that $\Phi = r_1$ and r_2 is in Φ_1. Thanks to proposition 2.9, we can write Φ as $\Phi_1; (\mathsf{cn}\, \Phi_1|_{K\{\rho\}}\, \mathsf{pr}\, \Phi_2); \Phi_2$. If the result of reducing r_2 in Φ_1 is Φ_1', then the result of reducing both redexes in any order is $\Phi_1'; K\{C|_\rho D\}; \Phi_2$.

The fact that a normal form is a synchronal derivation directly follows from definition 2.12.

■

Comment 6. The reduction of a redex doesn't create any new redex. As a consequence, one can obtain the synchronal form by reducing all the redexes in parallel.

Example 2.19. We show here how the sequential derivation in Example 2.15 can be rewritten to the synchronal derivation in the same example by several applications of the $\overset{\mathsf{sync}}{\to}$ rewriting. The sequential derivation is written as a composition of inference steps, which is possible by 2.9, before the $\overset{\mathsf{sync}}{\to}$ rewriting is applied in parallel.



Definition 2.20. Given a derivation Φ, we denote the normal form of Φ with respect to $\overset{\mathsf{sync}}{\to}$ by $\mathrm{sync}(\Phi)$.

Lemma 2.21. $\mathrm{sync}(\Phi;\Psi) = \mathrm{sync}(\Phi);\mathrm{sync}(\Psi)$.

Proof. We proceed by structural induction on Φ:

- the base case is trivial;
- $\mathrm{sync}((\Phi_1|_{K\{\rho\}}\Phi_2);\Psi)$
  $= \mathrm{sync}(\Phi_1|_{K\{\rho\}}(\Phi_2;\Psi))$
  $= \mathrm{sync}(\Phi_1);\mathrm{sync}(\mathsf{cn}\,\Phi_1|_{K\{\rho\}}\,\mathsf{pr}\,\Phi_2);\mathrm{sync}(\Phi_2;\Psi)$
  $= \mathrm{sync}(\Phi_1);\mathrm{sync}(\mathsf{cn}\,\Phi_1|_{K\{\rho\}}\,\mathsf{pr}\,\Phi_2);\mathrm{sync}(\Phi_2);\mathrm{sync}(\Psi)$
  $= \mathrm{sync}(\Phi);\mathrm{sync}(\Psi)$;
- when $\Phi = r(\Phi_1,\ldots,\Phi_n)$, we proceed by structural induction on Ψ
  - if $\Psi = \Psi_1|\Psi_2$ we argue similarly to the previous case;
  - if $\Psi = r(\Psi_1,\ldots,\Psi_n)$, we have that
    $\mathrm{sync}(\Phi;\Psi) = r(\mathrm{sync}(\Phi_1;\Psi_1),\ldots,\mathrm{sync}(\Phi_n;\Psi_n)) = \mathrm{sync}(\Phi);\mathrm{sync}(\Psi)$.

■

3. Atomic Flows

We now introduce a special kind of directed acyclic graphs, called *atomic flows*. Atomic flows associated with classical propositional derivations have been used to describe their normal forms, to define normalisation procedures on derivations and to prove properties

of these procedures. The atomic flow associated with a derivation represents the causal relationship between the creation and destruction of atoms in the derivation.

In this section we show that atomic flows can be associated with derivations in a logic-independent way, given certain very mild assumptions about the inference rules we use. We then show that atomic flows are invariants of the rewriting $\overset{\mathsf{sync}}{\to}$.

We first define atomic flows independently of derivations and deductive systems.

Definition 3.1. An *atomic flow* is a directed, acyclic graph $(V, E, \mathrm{up}, \mathrm{lo})$, such that

- V is a set of *vertices* and E a set of *edges*;
- $\mathrm{up}\colon E \to V \cup \{\top\}$ and $\mathrm{lo}\colon E \to V \cup \{\bot\}$ are, respectively, the *upper* and *lower* maps, where $\top, \bot \notin V$ and $\top \neq \bot$.

Atomic flows are denoted by ϕ and ψ (possibly with subscripts).
For every $\nu \in V \cup \{\top, \bot\}$, we define the set $L_\nu = \{\, \epsilon \mid \mathrm{up}(\epsilon) = \nu \,\}$ of *lower edges of* ν, the set $U_\nu = \{\, \epsilon \mid \mathrm{lo}(\epsilon) = \nu \,\}$ of *upper edges of* ν, and the set $E_\nu = L_\nu \cup U_\nu$ of *edges of* ν.
For an atomic flow ϕ, we call the set $U_\phi = L_\top$ (resp., $L_\phi = U_\bot$) the *upper* (resp., *lower*) *edges of* ϕ.

We can compose atomic flows similarly to how we compose derivations, and later we will see that the two notions work nicely together. We compose atomic flows by pairwise 'identifying' lower edges of one with upper edges of the other, according to a given one-to-one correspondence between the two.

Definition 3.2. Let $\phi_1 = (V_1, E_1, \mathrm{up}_1, \mathrm{lo}_1)$, $\phi_2 = (V_2, E_2, \mathrm{up}_2, \mathrm{lo}_2)$ be two atomic flows and f a bijection from a subset U'_{ϕ_2} of U_{ϕ_2} to a subset L'_{ϕ_1} of L_{ϕ_1}. The *composition* $\phi_1 ;_f \phi_2$ *of* ϕ_1 *and* ϕ_2 *with respect to* f is the flow $(V, E, \mathrm{up}, \mathrm{lo})$ defined as follows:

- the set of vertices V is the disjoint union of V_1 and V_2;
- the set of edges E is the disjoint union of E_1 and E_2, minus L_{ϕ_1};
- the up and lo maps of $\phi ;_f \psi$ are inherited from the correspondind maps of ϕ_1 and ϕ_2 in the obvious way, except that, for every $\epsilon \in U'_{\phi_2}$, we have $\mathrm{up}(\epsilon) = \mathrm{up}_1(f(\epsilon))$.

Atomic flows are top-down symmetric, and in the same way that derivations are 'flipped' in a negative context, we might also want to 'flip' atomic flows. We now define the flipping operation.

Definition 3.3. The *flipping* operator $\cdot^\perp$ on atomic flows is defined as follows: if $\phi = (V, E, \mathrm{up}, \mathrm{lo})$ an atomic flow, then $\phi^\perp$ is the atomic flow $(V, E, \mathrm{up}^\perp, \mathrm{lo}^\perp)$ where, for every $\epsilon \in E$, if $\mathrm{up}(\epsilon) = \top$ (resp., $\mathrm{lo}(\epsilon) = \bot$), then $\mathrm{up}^\perp(\epsilon) = \bot$ (resp., $\mathrm{lo}^\perp(\epsilon) = \top$), otherwise $\mathrm{up}^\perp(\epsilon) = \mathrm{lo}(\epsilon)$ (resp., $\mathrm{lo}^\perp(\epsilon) = \mathrm{up}(\epsilon)$).

In deep inference most common logics can be expressed using only atomic and linear inference rules. In that case, we are able to separate the logical from the structural content of derivations, and atomic flows represent the structural content.

For the rest of this section we fix a set of logical relations $\mathcal{R}$ and a set of of basic inference steps $\mathcal{S} = \mathcal{S}_a \cup \mathcal{S}_l$, where $\mathcal{S}_a$ is a set of instances of atomic rules, $\mathcal{S}_l$ is a set of instances of linear rules.

Atomic flow associated to a derivation

Intuitively, every vertex of the atomic flow corresponds to an atomic inference rule or its converse, and its incident edges to the atom occurrences of this inference rule.

If Φ is a derivation, then $oc(\Phi)$ is the set of occurrences of atoms in Φ. We define in the following the *enriched atomic flow* $(\mathit{fl}(\Phi), f^{\top}_{\Phi}, f^{\bot}_{\Phi})$ *of a derivation* $\Phi \in \mathcal{D}_{\mathcal{R},\mathcal{S}}$, where $\mathit{fl}(\Phi)$ is an atomic flow called the *atomic flow* of Φ and $f^{\top}_{\Phi} : oc(\mathsf{pr}(\Phi)) \to U_{\mathit{fl}(\Phi)}$ and $f^{\bot}_{\Phi} : oc(\mathsf{cn}(\Phi)) \to L_{\mathit{fl}(\Phi)}$ are bijections relating the upper (resp. lower) edges of the flow to the atom occurrences of the premiss (resp. conclusion) of the derivation.

(1) The enriched atomic flow $(\mathit{fl}(a), f^{\top}_{a}, f^{\bot}_{a})$ of an atom a is defined as follows: $\mathit{fl}(a)$ is the flow consisting of no vertex and one edge, and $f^{\top}_{a} = f^{\bot}_{a}$ is the bijection mapping the atom to the edge.

(2) The enriched atomic flow $(\mathit{fl}(\Phi), f^{\top}_{\Phi}, f^{\bot}_{\Phi})$ of a derivation $\Phi = r(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n)$ with $r \in \mathcal{R}_{m,n}$ is defined as follows: $\mathit{fl}(\Phi)$ is the disjoint union of $\mathit{fl}(\Phi_1)$, $\ldots$, $\mathit{fl}(\Phi_m)$, $\mathit{fl}(\Phi'_1)^{\bot}$, $\ldots$, $\mathit{fl}(\Phi'_n)^{\bot}$ and $f^{\top}_{\Phi}$ (resp., $f^{\bot}_{\Phi}$) is the disjoint union of $f^{\top}_{\Phi_1}$, $\ldots$, $f^{\top}_{\Phi_m}$, $f^{\bot}_{\Phi'_1}$, $\ldots$, $f^{\bot}_{\Phi'_n}$ (resp., $f^{\bot}_{\Phi_1}$, $\ldots$, $f^{\bot}_{\Phi_m}$, $f^{\top}_{\Phi'_1}$, $\ldots$, $f^{\top}_{\Phi'_n}$)

(3) The enriched atomic flow $(\mathit{fl}(\rho), f^{\top}_{\Phi}, f^{\bot}_{\Phi})$ of a basic inference step ρ is defined as follows:
- If ρ is an instance $C|D$ of a linear rule, then $\mathit{fl}(\rho) = \mathit{fl}(C)$, $f^{\top}_{\rho} = f^{\top}_{C}$ and $f^{\bot}_{\rho} = g \circ f^{\bot}_{C}$, where g is the bijection between the occurrences of atoms in D and the corresponding occurrences in C, which exits thanks to the linearity of the rule.
- If ρ is an instance $C|D$ of an atomic rule, then $\mathit{fl}(\rho) = (\{v\}, U_{\mathit{fl}(C)} + L_{\mathit{fl}(D)}, \mathrm{up}, \mathrm{lo})$, where up and lo are defined as follows:
  - $\mathrm{up}(e)$ is v, if $e \in U_{\mathit{fl}(C)}$ and $\top$, otherwise.
  - $\mathrm{lo}(e)$ is v, if $e \in L_{\mathit{fl}(D)}$ and $\bot$, otherwise.

  The flow ϕ is enriched by taking $f^{\top}_{\rho} = f^{\top}_{C}$ and $f^{\bot}_{\rho} = f^{\bot}_{D}$.

(4) The enriched atomic flow $(\mathit{fl}(\Phi), f^{\top}_{\Phi}, f^{\bot}_{\Phi})$ of a derivation $\Phi = \Phi_1|_{K\{\rho\}}\Phi_2$ is defined as follows. Suppose that ρ is $C|D$. We write Φ as $\Phi_1; K\{C\}|K\{D\}; \Phi_2$ if $K\{\ \}$ is positive and Φ as $\Phi_1; K\{D\}|K\{C\}; \Phi_2$ if $K\{\ \}$ is negative. We then obtain the flow of Φ by composing the flows of the compound derivations:
- $\mathit{fl}(\Phi) = \mathit{fl}(\Phi_1);_g \mathit{fl}(K\{C|D\});_h \mathit{fl}(\Phi_2)$, where $g = f^{\bot}_{\Phi_1} \circ (f^{\top}_{\mathit{fl}(K\{C|D\})})^{-1}$ and $h = f^{\bot}_{\mathit{fl}(K\{C|D\})} \circ (f^{\top}_{\Phi_2})^{-1}$;
- $f^{\top}_{\Phi} = f^{\top}_{\Phi_1}$
- $f^{\bot}_{\Phi} = f^{\bot}_{\Phi_2}$

Examples of atomic flows associated to derivations

We consider atomic flows for classical propositional derivations in SKS [GG08].

We first give the atomic flows associated to the basic inference steps associated to structural rules of SKS, from which the flows of all the SKS derivations are build. Vertices corresponding to the structural rules are represented by three different symbols (—, ▽ and △) and their incident edges (represented by vertical lines) correspond to the atom occurrences of the rules. The labels of the occurrences of atoms are also indicated on the edges they correspond to, in order to ease the reading.

$$\mathsf{ai}{\downarrow}\,\frac{}{a^{\mathbf{1}} \vee \bar{a}^{\mathbf{2}}} \;\to\; \mathbf{1}\sqcap\mathbf{2} \qquad \mathsf{aw}{\downarrow}\,\frac{}{a^{\mathbf{1}}} \;\to\; \mathbf{1} \qquad \mathsf{ac}{\downarrow}\,\frac{a^{\mathbf{1}} \vee a^{\mathbf{2}}}{a^{\mathbf{3}}} \;\to\; \mathbf{1}\curlyvee\mathbf{2}\,/\,\mathbf{3}$$

$$\mathsf{ai}{\uparrow}\,\frac{a^{\mathbf{1}} \wedge \bar{a}^{\mathbf{2}}}{} \;\to\; \mathbf{1}\sqcup\mathbf{2} \qquad \mathsf{aw}{\uparrow}\,\frac{a^{\mathbf{1}}}{} \;\to\; \mathbf{1} \qquad \mathsf{ac}{\uparrow}\,\frac{a^{\mathbf{3}}}{a^{\mathbf{1}} \wedge a^{\mathbf{2}}} \;\to\; \mathbf{3}\,/\,\mathbf{1}\curlywedge\mathbf{2}$$

Here is now the example of a flow associated to a general inference step, which consist of an application of a basic inference step in a context. The context creates edges not related to vertices.

$$\mathsf{ac}{\downarrow}\,\frac{a^{\mathbf{1}} \wedge \left[b^{\mathbf{2}} \vee \left[a^{\mathbf{3}} \vee a^{\mathbf{4}}\right]\right]}{a^{\mathbf{1}} \wedge \left[b^{\mathbf{2}} \vee a^{\mathbf{5}}\right]} \;\to\; \mathbf{1}\;\;\mathbf{2}\;\;\mathbf{3}\curlyvee\mathbf{4}\,/\,\mathbf{5}\;.$$

We finally give the example of a flow associated to to a sequential derivation.

$$\begin{array}{c}
\mathsf{ai}{\downarrow}\,\dfrac{\left(a^{\mathbf{1}} \wedge \left[\bar{a}^{\mathbf{3}} \vee \mathsf{t}\right]\right) \wedge \bar{a}^{\mathbf{8}}}{\left(a^{\mathbf{1}} \wedge \left[\bar{a}^{\mathbf{3}} \vee \left[\bar{a}^{\mathbf{4}} \vee a^{\mathbf{5}}\right]\right]\right) \wedge \bar{a}^{\mathbf{8}}} \\
= \dfrac{\left(a^{\mathbf{1}} \wedge \left[\left[\bar{a}^{\mathbf{3}} \vee \bar{a}^{\mathbf{4}}\right] \vee a^{\mathbf{5}}\right]\right) \wedge \bar{a}^{\mathbf{8}}}{} \\
\mathsf{s}\,\dfrac{\left[\left(a^{\mathbf{1}} \wedge \left[\bar{a}^{\mathbf{3}} \vee \bar{a}^{\mathbf{4}}\right]\right) \vee a^{\mathbf{5}}\right] \wedge \bar{a}^{\mathbf{8}}}{} \\
\mathsf{ac}{\downarrow}\,\dfrac{\left[\left(a^{\mathbf{1}} \wedge \bar{a}^{\mathbf{2}}\right) \vee a^{\mathbf{5}}\right] \wedge \bar{a}^{\mathbf{8}}}{} \\
\mathsf{ai}{\uparrow}\,\dfrac{\left[\mathsf{f} \vee a^{\mathbf{5}}\right] \wedge \bar{a}^{\mathbf{8}}}{} \\
= \dfrac{a^{\mathbf{5}} \wedge \bar{a}^{\mathbf{8}}}{} \\
\mathsf{ac}{\uparrow}\,\dfrac{\left(a^{\mathbf{6}} \wedge a^{\mathbf{7}}\right) \wedge \bar{a}^{\mathbf{8}}}{} \\
= \dfrac{a^{\mathbf{6}} \wedge \left(a^{\mathbf{7}} \wedge \bar{a}^{\mathbf{8}}\right)}{} \\
\mathsf{ai}{\uparrow}\,\dfrac{}{a^{\mathbf{6}} \wedge \mathsf{f}}
\end{array} \;\to\; \mathbf{1}\;\mathbf{3}\;\mathbf{8}\;\mathbf{4}\;\mathbf{5}\;\mathbf{2}\;\mathbf{7}\;\mathbf{6}$$

By design the composition of flows and the composition of derivations work together as expected:

Lemma 3.4. *For any atomic flows $\Phi\colon A \to B$ and $\Psi\colon B \to C$, we have that $\mathit{fl}(\Phi;\Psi) = \mathit{fl}(\Phi);_{f_\Phi^\perp \circ (f_\Psi^\top)^{-1}} \mathit{fl}(\Psi)$.*

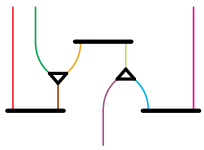
Atomics flows have been defined for sequential derivations in [GG08] and for synchronal derivations in [Gun09], in the case of classical propositionnal derivations in SKS. We now show that the two notions coincide in general, and that atomic flows are in fact invariants of the $\overset{\mathsf{sync}}{\to}$ rewriting.

Theorem 3.5. *If $\Phi \overset{\mathsf{sync}}{\to} \Psi$, then $\mathit{fl}(\Phi) = \mathit{fl}(\Psi)$.*

Sketch of proof. We first remark that $\mathit{fl}(K\{A'\}|_{K\{\rho\}}K\{B'\}) = \mathit{fl}(K\{A'|_\rho B'\})$. It then follows by the previous lemma that the flow of a synchronisation redex is the same as the flow of its contractum. The result is obtained by checking that the flow of a derivation is the same as the flow of the result of replacing a synchronisation redex by its contractum. ∎

Example 3.6. The sequential derivation on the left reduces to the synchronal derivation on the right, and they both have the atomic flow in the middle, where the correspondence between atom occurrences and edges is indicated by colours:

$$\begin{array}{l}
\mathsf{ai}{\downarrow}\dfrac{(a \wedge [\bar{a} \vee \mathsf{t}]) \wedge \bar{a}}{(a \wedge [\bar{a} \vee [\bar{a} \vee a]]) \wedge \bar{a}} \\
= \dfrac{(a \wedge [\bar{a} \vee [\bar{a} \vee a]]) \wedge \bar{a}}{(a \wedge [[\bar{a} \vee \bar{a}] \vee a]) \wedge \bar{a}} \\
\mathsf{s}\dfrac{(a \wedge [[\bar{a} \vee \bar{a}] \vee a]) \wedge \bar{a}}{[(a \wedge [\bar{a} \vee \bar{a}]) \vee a] \wedge \bar{a}} \\
\mathsf{ac}{\downarrow}\dfrac{[(a \wedge [\bar{a} \vee \bar{a}]) \vee a] \wedge \bar{a}}{[(a \wedge \bar{a}) \vee a] \wedge \bar{a}} \\
\mathsf{ai}{\uparrow}\dfrac{[(a \wedge \bar{a}) \vee a] \wedge \bar{a}}{[\mathsf{f} \vee a] \wedge \bar{a}} \\
= \dfrac{[\mathsf{f} \vee a] \wedge \bar{a}}{a \wedge \bar{a}} \\
\mathsf{ac}{\uparrow}\dfrac{a \wedge \bar{a}}{(a \wedge a) \wedge \bar{a}} \\
= \dfrac{(a \wedge a) \wedge \bar{a}}{a \wedge (a \wedge \bar{a})} \\
\mathsf{ai}{\uparrow}\dfrac{a \wedge (a \wedge \bar{a})}{a \wedge \mathsf{f}}
\end{array}$$

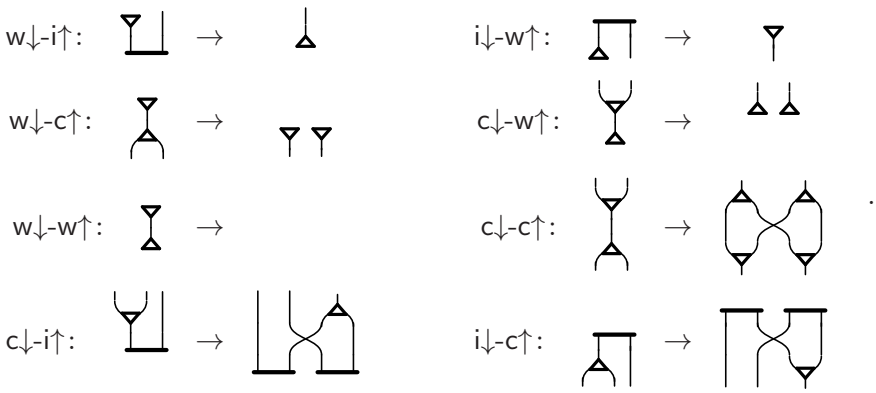


$$= \dfrac{\left(\mathsf{s}\dfrac{a \wedge \left[\bar{a} \vee \dfrac{\mathsf{t}}{\bar{a} \vee a}\right]}{\dfrac{a \wedge \dfrac{\bar{a} \vee \bar{a}}{\bar{a}}}{\mathsf{f}} \vee \dfrac{a}{a \wedge a}} \wedge \bar{a}\right)}{a \wedge \dfrac{a \wedge \bar{a}}{\mathsf{f}}} .$$

4. Atomic Flow Rewriting

We now give an example of the use of atomic flows with respect to the system SKS for propositional classical logic. We define reductions on flows and then we show how they can be lifted to derivations. This has been done for sequential derivations in [GG08] and for synchronal derivations in [Gun09]: here we show that the reductions on synchronal derivations correspond exactly to reductions on sequential derivations modulo $\overset{\mathsf{sync}}{\rightarrow}$.

Definition 4.1. We define graphical expressions of the kind $r\colon \phi' \rightarrow \psi'$, where r is a name and ϕ' and ψ' are flows:



Definition 4.2. For every expression $r\colon \phi' \rightarrow \psi'$ from Definition 4.1, the reduction $\rightarrow_r$ is defined, such that $\phi \rightarrow_r \psi$ if and only if ϕ' is a subflow in ϕ and we obtain ψ by replacing ϕ' with ψ' in ϕ, while respecting the correspondence of edges.

Theorem 4.3. *For each* $r \in \{\mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{c}{\uparrow}\}$ *and every* SKS*-derivation (resp., synchronal or sequential* SKS*-derivation)* $\Phi\colon A \rightarrow B$ *and every atomic flow* ψ*, such that* $\mathit{fl}(\Phi) \rightarrow_r \psi$*; there exists an* SKS*-derivation (resp., synchronal or sequential* SKS*-derivation)* $\Psi\colon A \rightarrow B$ *with flow* ψ*.*

Proof. We consider the case for $\mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}$, the other cases can be proven similarly. Assuming $\mathit{fl}(\Phi)$ contains

let every atom occurrence a in Φ that is mapped to the edge labelled with $\bullet$ be labelled $a^\bullet$. By Proposition 2.9 Φ must contain the two subderivations $\Phi' = \Phi'_1; \left(\mathsf{ac}{\downarrow}\,\dfrac{K_1\,[a \vee a]}{K_1\{a^\bullet\}}\right); \Phi'_2$ and $\Phi'' = \Phi''_1; \left(\mathsf{ac}{\uparrow}\,\dfrac{K_2\{a^\bullet\}}{K_2\,(a \wedge a)}\right); \Phi''_2$.

If Φ is synchronal, we have that $K_1\{\ \} = K_2\{\ \} = \{\ \}$ and we define

$$\hat{\Psi}' \;=\; \mathsf{m}\,\frac{\left[\dfrac{a}{a \wedge a} \;\vee\; \dfrac{a}{a \wedge a}\right]}{[a \vee a] \wedge [a \vee a]} \quad \text{and} \quad \hat{\Psi}'' \;=\; \left(\frac{a \vee a}{a} \;\wedge\; \frac{a \vee a}{a}\right) \quad .$$

Otherwise, we define

$$\hat{\Psi}' \;=\; \mathsf{ac}{\uparrow}\,\mathsf{m}\,\frac{\mathsf{ac}{\uparrow}\,\dfrac{K_1\,[a \vee a]}{K_1\,[(a \wedge a) \vee a]}}{K_1\,([a \vee a] \wedge [a \vee a])} \quad \text{and} \quad \hat{\Psi}'' \;=\; \mathsf{ac}{\downarrow}\,\mathsf{ac}{\downarrow}\,\frac{\dfrac{K_2\,([a \vee a] \wedge [a \vee a])}{K_2\,([a \vee a] \wedge a)}}{K_2\,(a \wedge a)} \quad .$$

Finally, we define

$$\Psi' = \Phi'_1; \hat{\Psi}'; \Phi'_2\{a^\bullet / ([a \vee a] \wedge [a \vee a])\} \quad \text{and} \quad \Psi'' = \Phi''_1\{a^\bullet / ([a \vee a] \wedge [a \vee a])\}; \hat{\Psi}''; \Phi''_2 \quad .$$

This allows us to obtain the derivation $\Psi\colon A \to B$ with the required atomic flow from Φ, by simultaneously applying the substitution $\{a^\bullet / ([a \vee a] \wedge [a \vee a])\}$, replacing Φ' with Ψ', and replacing Φ'' with Ψ''. ∎

Definition 4.4. Given $r \in \{\mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{c}{\uparrow}\}$, an SKS-derivation Φ, a flow ψ, such that $\mathit{fl}(\Phi) \to_r \psi$, and the SKS-derivation Ψ constructed in the proof of Theorem 4.3, we write $\Phi \to_r \Psi$.

Theorem 4.5. *Given* SKS*-derivations* Φ_1 *and* Φ_2*, such that* $\mathrm{sync}(\Phi_1) = \mathrm{sync}(\Phi_2)$*, then if* $\Phi_1 \to_r \Psi_1$ *and* $\Phi_2 \to_r \Psi_1$ *for some* $r \in \{\mathsf{w}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{w}{\downarrow}\text{-}\mathsf{w}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}, \mathsf{c}{\downarrow}\text{-}\mathsf{i}{\uparrow}, \mathsf{i}{\downarrow}\text{-}\mathsf{c}{\uparrow}\}$*, we have* $\mathrm{sync}(\Psi_1) = \mathrm{sync}(\Psi_2)$*.*

Sketch of proof. We sketch the proof for $r = \mathsf{c}{\downarrow}\text{-}\mathsf{c}{\uparrow}$, the other cases can be proved similarly: The result follows by Lemma 2.21 and the fact that

$$\frac{\dfrac{\dfrac{K_1\,[a \vee a]}{K_1\,[(a \wedge a) \vee a]}}{K_1\,[(a \wedge a) \vee (a \wedge a)]}}{K_1\,([a \vee a] \wedge [a \vee a])} \;\overset{\mathsf{sync}\star}{\to}\; K_1\left\{\frac{\dfrac{a}{a \wedge a} \;\vee\; \dfrac{a}{a \wedge a}}{[a \vee a] \wedge [a \vee a]}\right\} \quad \text{and}$$

$$\frac{\dfrac{K_2\,([a \vee a] \wedge [a \vee a])}{K_2\,([a \vee a] \wedge a)}}{K_2\,(a \wedge a)} \;\overset{\mathsf{sync}\star}{\to}\; K_2\left(\frac{a \vee a}{a} \;\wedge\; \frac{a \vee a}{a}\right) \quad .$$

∎

5. Conclusions

We have seen, in this paper, that we can reduce the syntactic bureaucracy of proof systems using a logic-independent formalism, without sacrificing the usual proof-theoretic tools and properties like cut-elimination and complexity. We eliminate 'type A' bureaucracy, i.e., the irrelevant order of application of two rules to two independent subformulae [Gug04a].

The formalism here introduced, called *open deduction*, generalises the calculus of structures, which in turn generalises the sequent calculus and natural deduction. This generality does not claim a price on the naturalness of the formalism: it simply extends to proofs the structure of formulae, that is only partly used in the sequent calculus.

However, the necessary technology to deal with cut elimination is not trivial, and only now we are in a position to deal with it, relying on recent developments of the calculus of structures. This paper shows that if we can normalise a proof in the calculus of structures, then we can eliminate its bureaucracy in open deduction. Moreover, since the atomic flow of proofs is invariant under bureaucracy elimination, we can start developing atomic-flow-based cut elimination directly in open deduction.

In a forthcoming paper, we will define an even more general formalism, containing open deduction and eliminating a further type of bureaucracy, namely the irrelevant order of application of two rules to two nested subformulae. This is dubbed 'type B' bureaucracy in [Gug04b]. The reason to separate, at this stage, open deduction and its further generalisation, is that addressing type B bureaucracy requires a level of abstraction that would be best supported by further developments of atomic flows, to which our efforts are dedicated.

References

[BGGP09] Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A quasipolynomial cut-elimination procedure in deep inference via atomic flows and threshold formulae. Submitted. `http://cs.bath.ac.uk/ag/p/QPNDI.pdf`, 2009.

[Brü03] Kai Brünnler. Two restrictions on contraction. *Logic Journal of the IGPL*, 11(5):525–529, 2003. `http://www.iam.unibe.ch/~kai/Papers/RestContr.pdf`.

[Brü04] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. Logos Verlag, Berlin, 2004. `http://www.iam.unibe.ch/~kai/Papers/phd.pdf`.

[BT01] Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *Lecture Notes in Computer Science*, pages 347–361. Springer-Verlag, 2001. `http://www.iam.unibe.ch/~kai/Papers/lcl-lpar.pdf`.

[GG08] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008. `http://www.lmcs-online.org/ojs/viewarticle.php?id=341`.

[Gug] Alessio Guglielmi. Deep inference. Web site at `http://alessio.guglielmi.name/res/cos`.

[Gug04a] Alessio Guglielmi. Formalism A. `http://cs.bath.ac.uk/ag/p/AG11.pdf`, 2004.

[Gug04b] Alessio Guglielmi. Formalism B. `http://cs.bath.ac.uk/ag/p/AG13.pdf`, 2004.

[Gug07] Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, 2007. `http://cs.bath.ac.uk/ag/p/SystIntStr.pdf`.

[Gun09] Tom Gundersen. *A General View of Normalisation Through Atomic Flows*. PhD thesis, University of Bath, 2009.

[Str09] Lutz Straßburger. From deep inference to proof nets via cut elimination. *Journal of Logic and Computation*, 2009. In press. `http://www.lix.polytechnique.fr/~lutz/papers/deepnet.pdf`.